AMENDMENTS TO THE SPECIFICATION

IN THE SPECIFICATION:

Please replace paragraph [0009] with the following amended paragraph:

In one version of the invention, the memory manager responds to a warm-reboot by providing a set of data to an application during the warm reboot. The set of data has been previously indicated by the application to be sent to ~~it~~ the application in the event of a warm-reboot from the warm-reboot persistent memory. In one embodiment, the application may indicate a particular state during a warm-reboot in which the data is to be sent.

Please replace paragraph [0010] with the following amended paragraph:

In another version of the invention, the memory manager responds to a cold-reboot by providing a set of data previously indicated by an application to be sent to it in ~~by~~ the event of a cold-reboot from the cold-reboot persistent memory to the application during the cold-reboot. In one embodiment, the application may indicate a particular state during a cold-reboot in which the data is to be sent.

Please replace paragraph [0028] with the following amended paragraph:

Figure 12 illustrates an embodiment ~~of a~~ ~~1200~~ of storage of data in cold-reboot persistence memory by a memory manager in accordance with the present invention.

Please replace paragraph [0032] with the following amended paragraph:

In one embodiment, a node is a data and telecommunications transport platform. Examples of activities which processing at a node may comprise include routing and add/drop multiplexing. One type of a node is an optical node that performs activities for processing the optical signals received, transmitted, and passed through ~~at~~ the node. An optical node may process one or more wavelengths in the optical filters, that pass through wavelengths that are not added or dropped at the node. A node may also perform optical-to-electrical and electrical-to-optical conversions as part of the processing of wavelengths. For example, when dropping a wavelength from the network to a tributary node, optical-to-electrical conversion may be necessary because of only electrical equipment being on the tributary side or as part of wavelength conversion to a wavelength that can be processed by the tributary node.

Please replace paragraph [0034] with the following amended paragraph:

An example of a sub-system having a memory management system according to the invention is a node element module of a node. A typical node in a network typically comprises node element modules for performing functions, examples of which are payload traffic-carrying functions or maintenance functions. In one embodiment, the node element module comprises an optical element, and the optical element is connected with electronic hardware and/or software for controlling or monitoring the optical element.

Please replace paragraph [0038] with the following amended paragraph:

The processor module 106 also has an intra-nodal communication link 128A to ~~an embodiment of a~~ another node element module, the primary signaling channel module 120A and another intra-nodal communication link 128B to a back-up signaling channel module 120B. The persistent storage module 104 also has an intra-nodal communication link 126A to the primary signaling channel module 120A and another intra-nodal communication link 126B to the back-up signaling channel module 120B. Each of the ~~other~~ node element modules in the transport complex has a communication link 130A, 132A, 134A to the primary signaling channel module 120A. Each node element module on the transport complex is also communicatively

coupled via a communication link 130B, 132B, 134B to the back-up signaling channel module 120B.

Please replace paragraph [0040] with the following amended paragraph:

In this embodiment, the control subsystem 144 comprises an architecture, which includes a processor 123, program storage memory 122, execution memory 124, and flash memory 136. The control subsystem 144 may be augmented with additional input/output (I/O) and control capabilities, as appropriate.

Please replace paragraph [0041] with the following amended paragraph:

The program storage memory comprises executable software applications 131, 135 including a reboot state machine ~~121~~ application 121 and a memory manager ~~138~~ application 138. The applications are executed by the processor 123 which is communicatively coupled to access the program storage memory 122, flash memory 136, and execution memory 124. The processor 123 is also communicatively coupled to the control unit 162 associated with the optical node element 160. In the embodiment shown, the flash memory 136 comprises non-volatile cold-reboot persistence memory 125. The execution memory 124 comprises a warm-reboot persistence memory which may be embodied as random access memory

(RAM) 127. In the embodiment of Figure 3, the operating system (not shown does not clear the contents of the persistence memory during a reboot.

Please replace paragraph [0042] with the following amended paragraph:

Figure 4A illustrates an embodiment of a memory system 400 in accordance with the present invention. The system 400 comprises a memory manager 138, a warm-reboot persistence memory 127, and a cold-reboot persistence memory 125. The warm-reboot persistence memory 127 stores sets of data that must persist over warm-reboots. The warm-reboot persistence memory 127 may be embodied in random access memory (RAM). The non-volatile cold-reboot persistence memory 125, such as may be embodied in Flash memory, stores sets of data that must persist over cold-reboots. In the example shown, the warm-reboot persistence memory 127 includes a first memory region 335 marked or designated active ~~335~~ and a second memory region 337 marked or designated alternate ~~337~~. The cold-reboot persistence memory 125 also comprises a first memory region 342 marked or designated active ~~342~~ and a second memory region 344 marked or designated alternate ~~344~~. In one embodiment, the designation or marking of a region may be changed from active to alternate and vice versa responsive to events.

Please replace paragraph [0046] with the following amended paragraph:

Those of ordinary skill in the art recognize that the systems and methods of the invention may be embodied in software and/or hardware embodied within one or more computer usable ~~mediums~~ media. An example of a computer usable medium is a memory. For example, the memory manager 138 may be software embodied in program storage memory 122. Furthermore, it ~~is~~ will be understood by those of skill in the art that the various embodiments of the systems and methods of the invention may be implemented in hardware, software, firmware or any combination of these. Additionally, those skilled in the art will appreciate that although the modules are depicted as individual units, the functionality of the modules may be implemented in a single unit (e.g. a single software program) or any combination of units.

Please replace paragraph [0047] with the following amended paragraph:

Figure 4B illustrates an embodiment of a warm-reboot persistence memory 127 as random access memory 127 (RAM) ~~127~~. In the embodiment shown, the RAM ~~127~~ includes the active region 335, having an active region header 414 and active region data portion 404, and the alternate region 337 having an alternate region header 416 and an alternate region data portion 410. In this embodiment,

the content of the alternate region 337 is illustrated as a mirror image of the content of the active region 335.

Please replace paragraph [0048] with the following amended paragraph:

Each of the active data portion 404 and the alternate data portion 410 further comprise one or more sets 402, 406, 408, 412 of data. In this embodiment, each set of data has a data header 436, 440, 444, 448 and a data portion 438, 442, 446, 450. In one embodiment, the data header 436 will store the application identifier, including an endpoint identifier (ID), object ID and data key, as well as data size and the reboot state indicator for its respective set of data.

Please replace paragraph [0052] with the following amended paragraph:

Figure 5 illustrates an embodiment of an overall method 500 of storage of data by a memory manager in accordance with the present invention. A system such as the embodiment shown in Figure 4A may perform the embodiment of the method illustrated in Figure 5. For purposes of discussion, the method of Figure 5 will be discussed in terms of the embodiments of Figures 4A, 4B and 4C. An application determines which set of data is a set of data that is to be saved over either a warm-reboot or a cold reboot. An application ~~sends~~

~~502~~ sends, in step 502, a message to the memory manager application 138. The message 314 requests storage of a set of data in reboot persistence memory. The message in this embodiment includes a set of data to be stored in memory, an application identifier for the set of data, and reboot state indicator. The message handler 310 of the memory manager 138 receives ~~504~~ (in step 504) the message 314 from the application. The set of data and the application identifier are sent to the memory read/write module 306. The memory read/write module 306 determines ~~506~~ (in step 506) whether reboot persistence memory has been allocated for the set of data. Responsive to memory not having been allocated, a portion the region of reboot persistence memory is allocated ~~510~~ (in step 510) for the set of data. An example of a portion of the region would be a block of memory. The memory manager 138 associates ~~514~~ (in step 514) the application identifier with the portion of reboot persistence memory in which the set of data is stored. Responsive to memory having been allocated, the set of data is stored ~~512~~ (in step 512) in the portion of reboot persistence memory associated with the application identifier.

Please replace paragraph [0054] with the following amended paragraph:

Figure 6 illustrates an embodiment of a method 612 for determining the validity of the application data being stored in a

reboot persistence memory. A system such as the embodiment shown in Figure 4A may perform the embodiment of the method illustrated in Figure 6. For purposes of discussion, the method of Figure 6 will be discussed in terms of the embodiment of Figure 4A. As shown in the embodiment of Figures 4B and 4c, the active and alternate regions are mirror images of each other. The designations or markings as alternate and active can be changed. In one embodiment, the marking for the region is stored in the region's header. The memory read/write module 306 writes 622 the data to the second memory region 337,344 marked alternate ~~337, 344~~. The memory module calculates 624 a cyclic redundancy check value ~~on~~ for the second memory region marked alternate. A cyclic redundancy check value may be used to test the integrity of the data. The cyclic redundancy check value in the header 416, 428 of the second memory region 337, 344 marked alternate is updated 626. The marking of the second memory region 337, 344 is changed 628 from alternate to active. The marking of the first memory region 335, 342 is changed 630 from active to alternate. The data is copied 632 from the second memory region 337, 344 now marked active ~~337, 344~~ to the first memory region 335,342 now marked alternate ~~335, 342~~.

Please replace paragraph [0055] with the following amended paragraph:

Figure 7A illustrates an embodiment of a method 700 for providing application data from warm-reboot persistence memory to the application during a warm-reboot of a circuit card such as the embodiment node of an element module. The embodiment of a method illustrated in Figure 7A may be performed by a system such as the embodiment of Figure 4A. For purposes of discussion, the method of Figure 7A will be discussed in terms of the embodiments of Figures 3 and 4A. Referring back to Figure 3, the embodiment of the node element module includes a reboot state machine application 121. In this embodiment, the reboot state machine application sends messages to the memory manager 138 to indicate that a warm-reboot has occurred and the current state of the warm-reboot. Examples of states of either a warm-reboot or a cold-reboot are an initialization state, an unprovisioned state, a provisioning state, a pre-running state or a running state. The message handler 310 receives 702 the indication in a message that a warm-reboot of the node element module has occurred. The indication in this example also includes a state of the reboot, for example the initialization state. In the context of Figure 4A, the message handler 310 of the memory manager receives subsequent indications of transitions between states of the warm-reboot of the node element module. Under the control of the warm-reboot module 312, the memory read/write module 306 cycles through the warm-reboot persistence memory 127 and retrieves 704 the one or more sets of data indicated to be sent

to an application during a particular state during the warm-reboot. The memory manager 138 provides 706 each set of data from the warm-reboot persistence memory to the application during the state indicated by the stored reboot state indicator for the set of data. In this way, the application does not have to send a message during the reboot state in order to retrieve its data. The data is sent by the memory manager 138 automatically. In one embodiment, a message including the set of data is sent during the previously indicated reboot state to the application. In another embodiment, in which the set of data is particularly large, the message to the application may include a pointer to a memory location from which the data may be read.

Please replace paragraph [0058] with the following amended paragraph:

Figure 8A illustrates an embodiment of ~~an~~ a method for providing application data from non-volatile memory to the application during a cold-reboot of a circuit card such as the embodiment of a node element module illustrated in Figure 3. The embodiment of a method illustrated in Figure 8A may be performed by a system such as the embodiment of Figure 4A. For purposes of discussion, the method of Figure 8A will be discussed in terms of the embodiment of Figure 4A. The message handler 310 of memory manager 138 receives 802 an indication, such as in a message, that

cold-reboot of the node element module has occurred. The indication in this example also includes a state of the reboot, for example the initialization state. The message handler 310 receives subsequent indications of transitions between states of the cold-reboot of the node element module. Under the control of the cold-reboot module 308, the memory read/write module 306 cycles through the non-volatile cold-reboot persistence memory 125 and retrieves 804 the one or more sets of data indicated previously, such as by the stored reboot state indicator for each set, to be sent to an application during a particular state during the cold-reboot. The memory manager 138 provides 806 each set of data from the cold-reboot persistence memory to the application during the state indicated by the stored reboot state indicator for the set of data. In this way, the application does not have to send a message during the reboot state in order to retrieve its data. The data is sent by the memory manager 138 automatically. In the example of Figure 2, if each node element module has its own associated cold-reboot persistence storage memory associated with it, each node element module can store its own configuration information to be saved over a cold-reboot. This further decreases messaging with other modules such as the administrative node processor module 106 of the administrative complex 102 resulting in decreased reboot time. In one embodiment, a message including the set of data is sent during the previously indicated reboot state to the application. In

another embodiment, in which the set of data is particularly large, the message to the application may include a pointer to a memory location from which the data may be read.

Please replace paragraph [0059] with the following amended paragraph:

Figure 8B illustrates an embodiment of a method 820 for determining validity of the application data being retrieved from cold-reboot persistence memory during a cold-reboot. A system such as the embodiment shown in Figure 4A may perform the embodiment of the method illustrated in Figure 8B. For purposes of discussion, the method of Figure 8B will be discussed in terms of the embodiment of Figure 4A. Responsive to a cold-reboot of the node element module, the memory manager 138 performs 822 an integrity test, such as a cyclic redundancy check, for the active region 342 marked in the non-volatile cold-reboot persistence memory region. A determination is made 823 whether the integrity test indicated valid data. Responsive to a successful integrity test result indicating valid data for the active region, each set of data is provided 824 from the active region to the application indicated, in its respective application identifier during the cold-reboot as indicated in the reboot state indicator for the set. The reboot state indicator may indicate simply either a warm or cold-reboot or a particular state of either kind of reboot. Because the

application identifier and reboot state indicator have been stored prior to the cold-reboot for the set of data, it is not necessary for the application to request the data during the reboot.

Please replace paragraph [0066] with the following amended paragraph:

Each embodiment of a signaling channel module card in Figure 10 comprises an optical transceiver unit 182, 182', an intra-nodal communication unit 180, 180', and a control subsystem 144', 144". The optical transceiver unit comprises an optical transmitter such as a laser, an optical receiver and related control logic for the transmitter and receiver. The optical transceiver unit 182, 182' receives and transmits an optical signaling channel for the transfer of maintenance or supervisory information between a node and one or more other nodes in the optical network.

Please replace paragraph [0071] with the following amended paragraph:

In the context of Figure 10, the message from fault correlator copy 1 is forwarded by the processor 123' to the intra-nodal communication unit 180 and transferred through a communication link ~~196~~ 194 to the intra-nodal communication unit 180' of the back-up card where the processor 123" of the back-up card forwards it to the message handler (see 310, Figure 4A) of the memory manager 138"

of the back-up card. The memory manager 138" receives 904 the message from fault correlator copy 1 (step 909, Fig. 9) and stores 906 the set of data in the reboot persistence memory (127" or 125") located on the back-up signaling channel module card 120B as indicated by the reboot state indicator for the set. The manager 138" associates the set of data with the copy of the application (e.g. fault correlator copy 2) located on the second card. The fault correlator copy 1 will also typically request storage of the set of data to the memory manager 138' on its own card for storage.

Please replace paragraph [0072] with the following amended paragraph:

In the embodiment of Figure 9B, the memory manager 138' of the first card, the primary signaling channel module card, sends 910 a message (see Figure 10, 190 for illustrative purposes of an information flow) to the memory manager 138" on the second node element card, back-up signaling channel module card 120B. The message requests storage of a set of data in a reboot memory (127" or 125") on the second card. The memory manager 138" receives 914 the message from fault correlator copy 1 and stores 916 the set of data in the persistence memory, either random access memory 127" or cold-reboot persistence memory 125" located on the back-up signaling channel module card 120B in accordance with the reboot state indicator. The manager 138" stores the set of data on the

second card in memory associated with the application identifier. In this way, the set of data is associated with the copy of the application (e.g. fault correlator copy 2) located on the second card.

Please replace paragraph [0073] with the following amended paragraph:

Figure 11 illustrates an embodiment of a method 1100 for adjusting for a new version of a set of data in reboot memory. A new version often results in a data size change in the set of data. A system such as the embodiment shown in Figure 4A may be used to implement the embodiment of the method of Figure 11. The method of Figure 11 will be discussed on the context of Figure 4A for illustrative purposes. The message handler 310 determines 1101 that a change in data size for a set of data has occurred. For example, the message may include a data size indicator indicating the amount of data to be stored, and the message handler compares the data size indicated with that previously stored for this set of data. The data size change module 304 may reinitialize 1102 the second memory region 337, 344 marked alternate ~~337, 344~~ to clear the current region. The data size change module 304 instructs the memory read/write module 306 to overwrite 1104 the second memory region 337, 344 marked alternate ~~337, 344~~ with the data in the first memory region 335, 342 marked active ~~335, 342~~ up to the

location associated with the set of data that has had a change in data size. Under the control of the data size change module 304, the memory read/write module 306, at the location in the second memory region marked alternate for this set of data with its changed data size writes 1106 a new version of this set of data with its changed data size. Then the data in the first memory region 335, 342 marked active ~~335, 342~~ that comes after the set of data that has had a data size change is written 1108 to the second memory region 337, 344 marked alternate ~~337, 344~~. The data size change module causes the second memory region 337, 344 marked alternate ~~337, 344~~ to be marked 1110 as active. The first memory region 335, 342 marked active ~~335, 342~~ is now marked 1112 alternate. The first memory region marked alternate is reinitialized 1114. The data from the second memory region marked active is copied 1116 to the first memory region marked alternate, In this way, the memory manager 138 controls the updating of the reboot memory without the need for conversion routines being performed by the other applications.